Protecting Information Assets
- Unit 5b -

Application Security

Agenda

- Introduction
- Software development life cycle (SDLC)
- SDLC and security
- Test taking tip
- Quiz

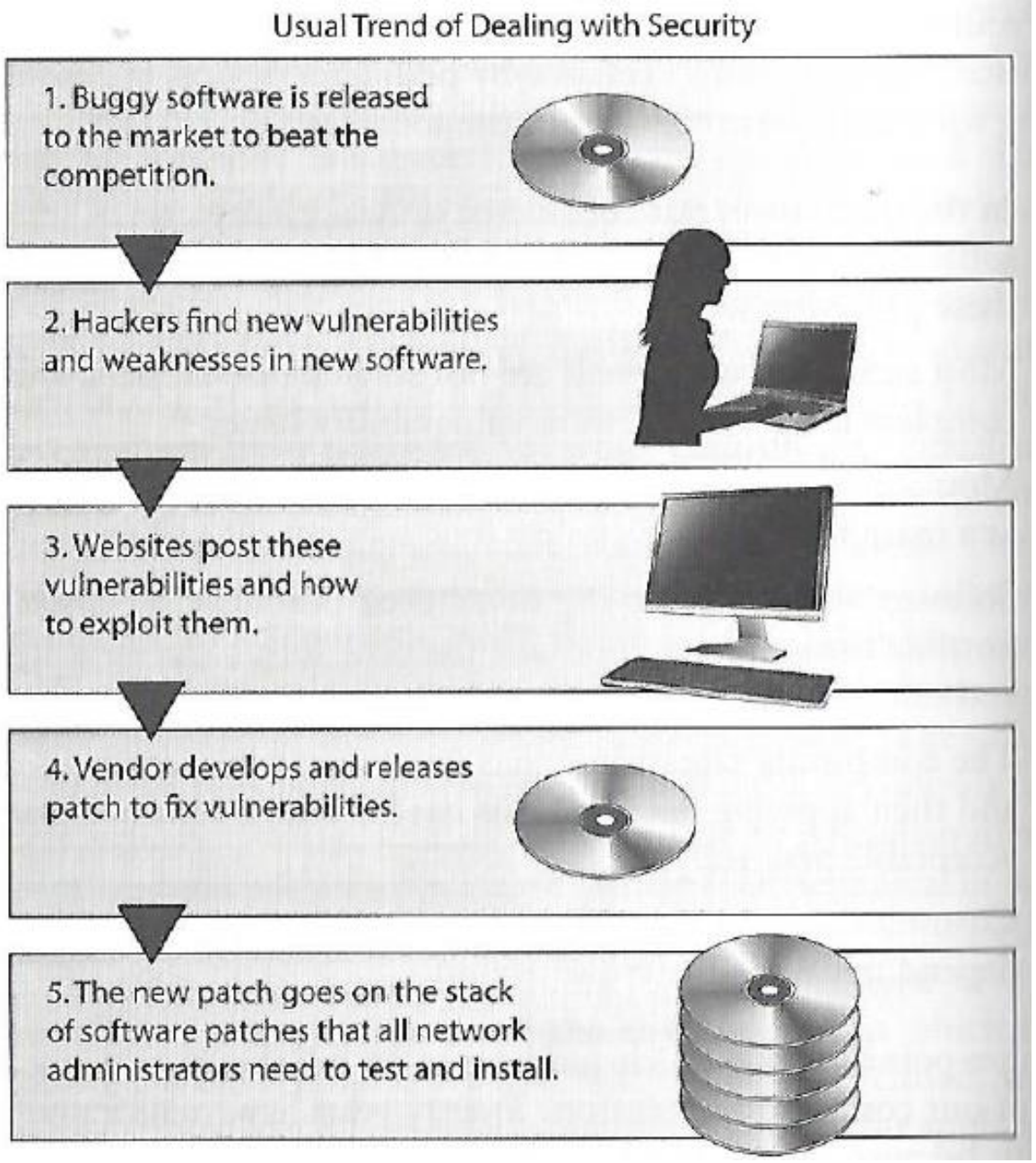
Application Security

As applications become more accessible though the web, cloud and mobile devices,

organizations are being forced to abandon their reactive approach to security and, instead,

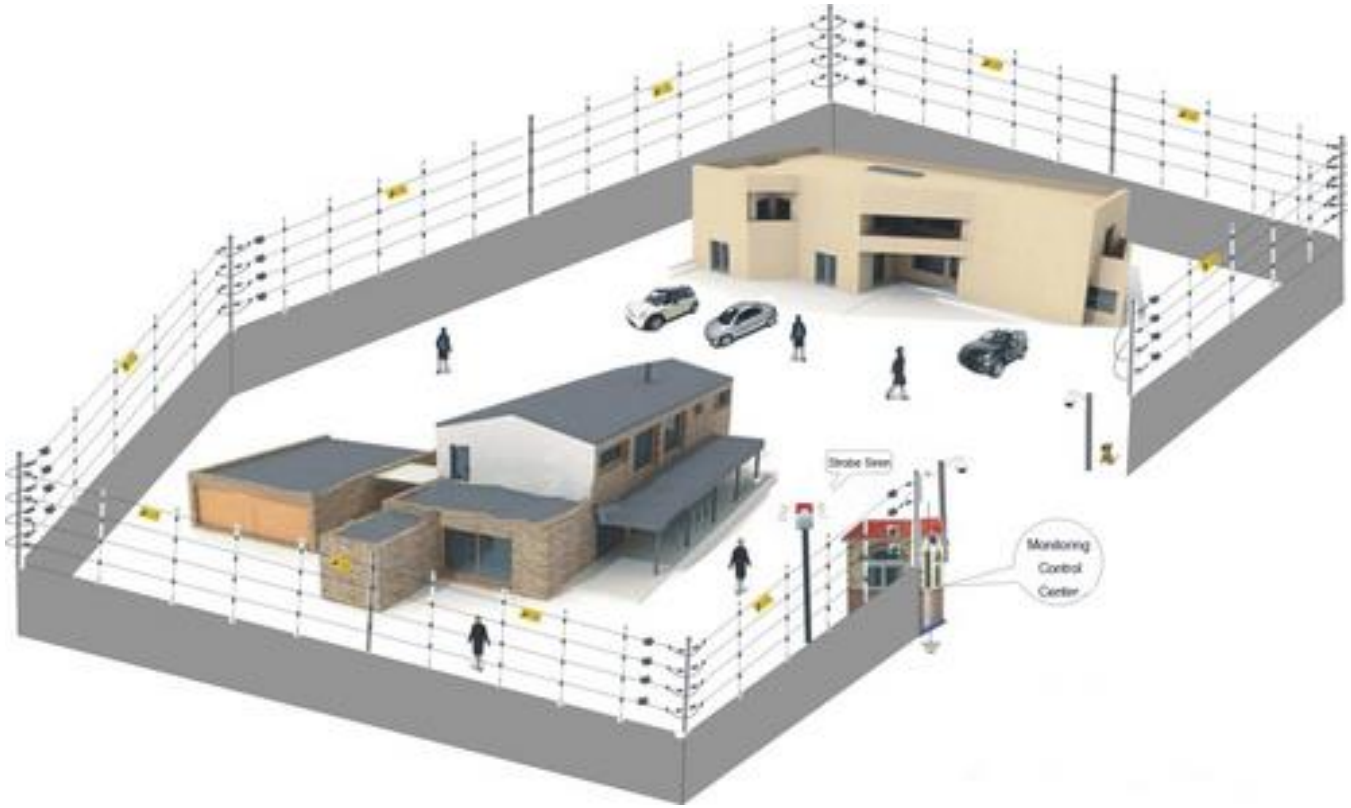
to take a proactive approach by minimizing risk directly in the software they buy, create and use to serve themselves and their customers

Usual trend

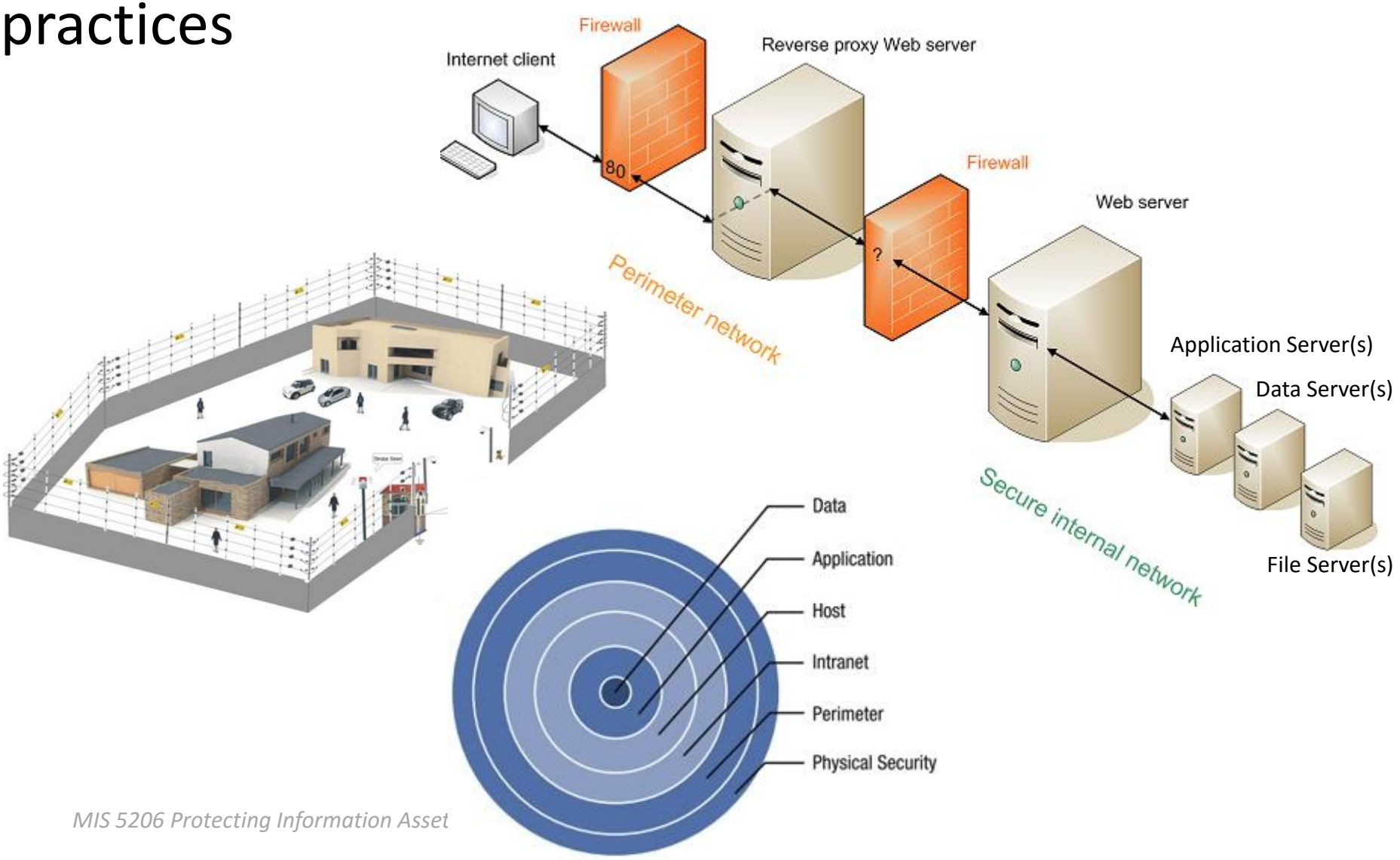


Harris, S. and Maymi F. (2016) CISSP All-in-One Exam Guide, McGraw-Hill Education, p. 1080

Perimeter security solutions are often relied on as a solution to insecure application development practices



Perimeter security solutions are often relied on as a solution to insecure application development practices

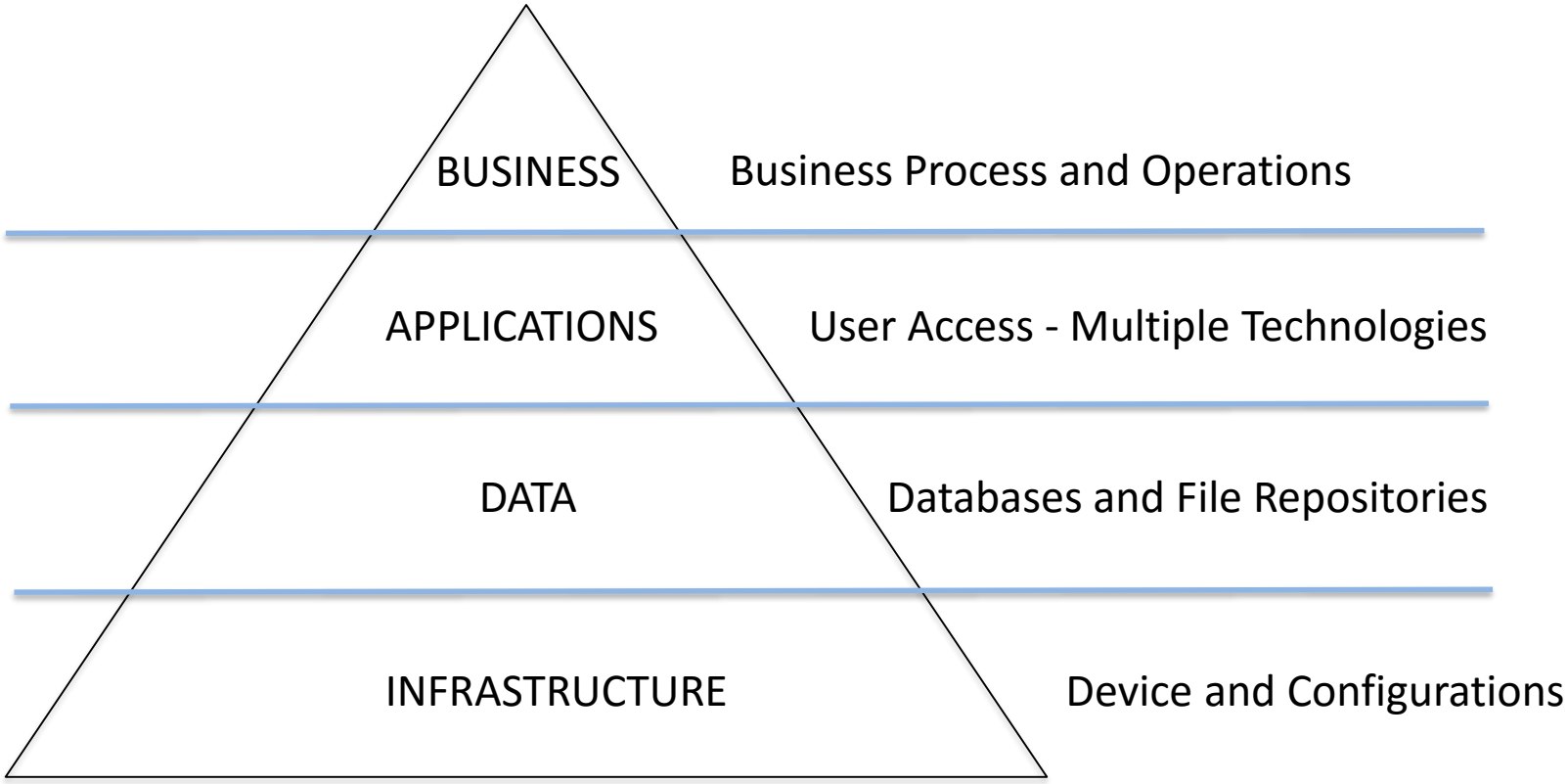


Past and current situation….

- Application developers are not security professionals
  - *Software vendors skip proper security architecture, design and testing steps as they race to beat competitors to market with new features*
- Secure application development practices have not historically been taught in computer science and other academic departments, and are only recently being considered and adopted by developers
- Development projects' scope and budgets focus on functionality, not security
- Security professionals typically not software developers
  - Often lack insight for understanding of software vulnerabilities
- **IT customers…**
  - "Trained" to expect to receive flawed software needing upgrades and patches
  - ***Unable to control flaws in software they purchase, so they rely on perimeter protection***

Security Architecture

Security strategy needs to be a consideration at each level of the architecture



Best Practice: Build Security In

Security Architecture	Creation, use and enforcement of System Architecture standards provides the basic building blocks for developing, implementing and maintaining secure applications
Software Development Life Cycle	Attention to security throughout the Software Development Life Cycle (SDLC) is the key to creating secure, manageable applications regardless of platform or technologies
Procurement Standards	Describing the process and detailed criteria that will be used to assess the security level of third party software enables companies to make strategic, security-sensitive decisions about purchased software purchases

Software Development Life Cycle

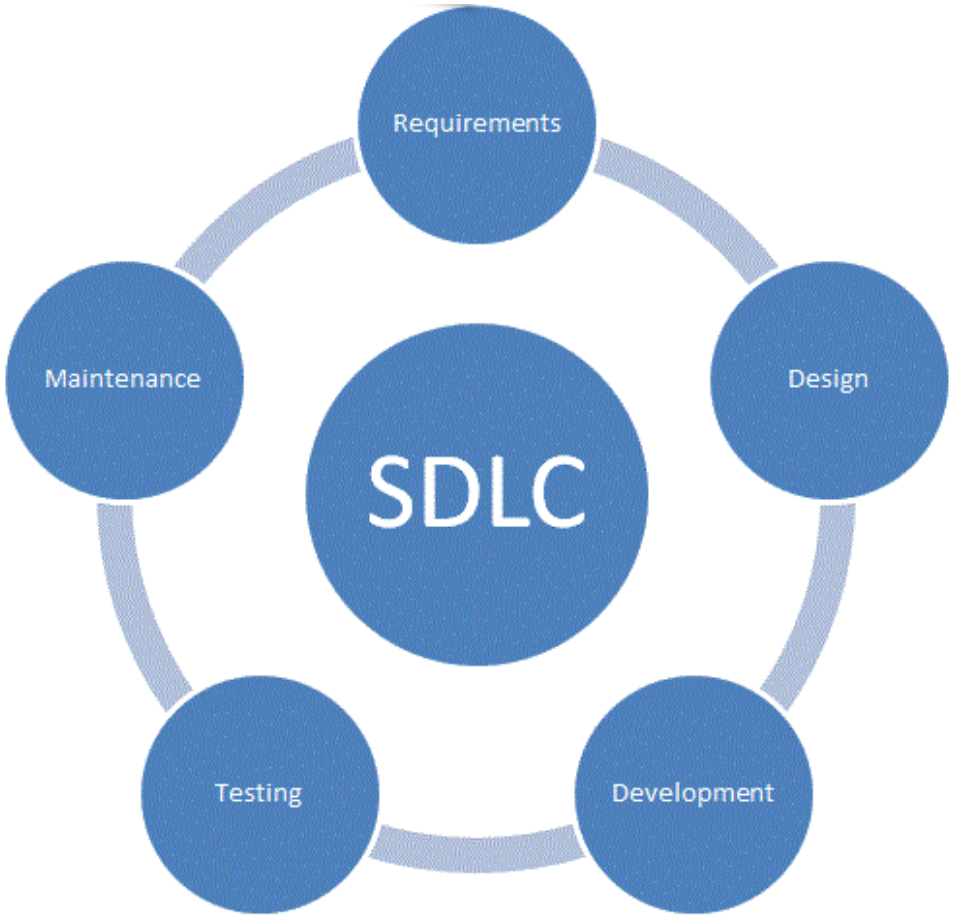
Requirements
- Why the software was created (i.e. goals)
- Who the software was created for
- What the software is intended to do

Design
- Specifications identifying how software and data will be formed to accomplish goals and used to meet requirements

Development
- Programming software code implemented and integrated to meet specifications

Testing-Validation
- Assuring software and data works as planned to meet the goals

Release-Maintenance
- Deploying software and data, and assuring they are properly configured, patched and monitored



Software Development Life Cycle (SDLC)

1. **Requirements analysis**
2. **Design**
3. **Develop (“*make*”) / Implement (“*buy*”)**
4. **Testing/Validation**
5. **Release/Maintenance**

Software Development Life Cycle (SDLC)

1. **Requirements analysis**
   - *Informational, functional, behavioral, and performance specifications…*
2. **Design**
   - *Data models and data dictionary, work process and status transition models, input/output models, data flow models, flow of control models…*
3. **Develop (“*make*”) / Implement (“*buy*”)**
   - *Source code control system, code reviews, daily builds, automated CASE tools…*
4. **Testing/Validation**
   - *Unit testing and integration testing (daily builds), manual and regression testing, user acceptance testing*
5. **Release/Maintenance**
   - *Release testing*

SDLC and Security

1. **Requirements analysis**
   - *Informational, functional, behavioral, and performance specifications…*

   + CIA risk assessment, + Risk-level acceptance,…
2. **Design**
   - *Data models and data dictionary, work process and status transition models, input/output models, data flow models, flow of control models…*

   + Threat modeling, + Attack surface analysis,…
3. **Develop (“*make*”) / Implement (“*buy*”)**
   - *Source code control system, code reviews, daily builds, automated CASE tools…*

   + Developer security training, + Static analysis, + Secure code repositories,…
4. **Testing/Validation**
   - *Unit testing and integration testing (daily builds), manual and regression testing, user acceptance testing*

   + Dynamic analysis, + Fuzzing,…
5. **Release/Maintenance**
   - *Release testing*

   + Separation of duties, +Change management,…

SDLC and Security

Requirements analysis

– *Informational, functional, behavioral, and performance specifications…*

+ CIA risk assessment, + Risk-level acceptance,…

Organisation & relevant process		Information Asset Details									
Operating Unit / Function	**Process name**	**Name of Asset**	**Personal Identifiying Information (PII)** (Y/N)	**Personal Health Information (PHI)** (Y/N)	**Critical Infrastructure Information (CII)** (Y/N)	**Customer Data** (Y/N)	**Organization Data** (Y/N)	**Confidentiality**	**Integrity**	**Availability**	**Categorization**
Thermal Distribution Syster	ChilledWater	TDS	N	N	Y	N	Y	Low	Medium	Medium	
Thermal Distribution Syster	HeatedWater	TDS	N	N	Y	N	Y	Low	Medium	Medium	
Thermal Distribution System		**TDS**	N	N	Y	N	Y	Low	Medium	Medium	**Medium**
Communication	Data	COM	N	N	Y	N	Y	Medium	Medium	Medium	
Communication	Voice	COM	N	N	Y	N	Y	Medium	Medium	Medium	
Communication	Security	COM	N	N	Y	N	Y	High	High	High	
Communication		**COM**	N	N	Y	N	Y	High	High	High	**High**
Public Works	Sewer	Utilities	N	N	Y	N	Y	Low	Medium	Low	
Public Works	Stormwater	Utilities	N	N	Y	N	Y	Low	Medium	Low	
Public Works	Water	Utilities	N	N	Y	N	Y	Low	Medium	Low	
Public Works		**Utilities**	**N**	**N**	**Y**	**N**	**Y**				**Medium**
External	**Parcels**	Parcels	Y	N	N	Y	N	Low	Low	Low	Low

Software requirements often specified with…

1. **Information model** – Type and content of information that will be processed and how it will be processed

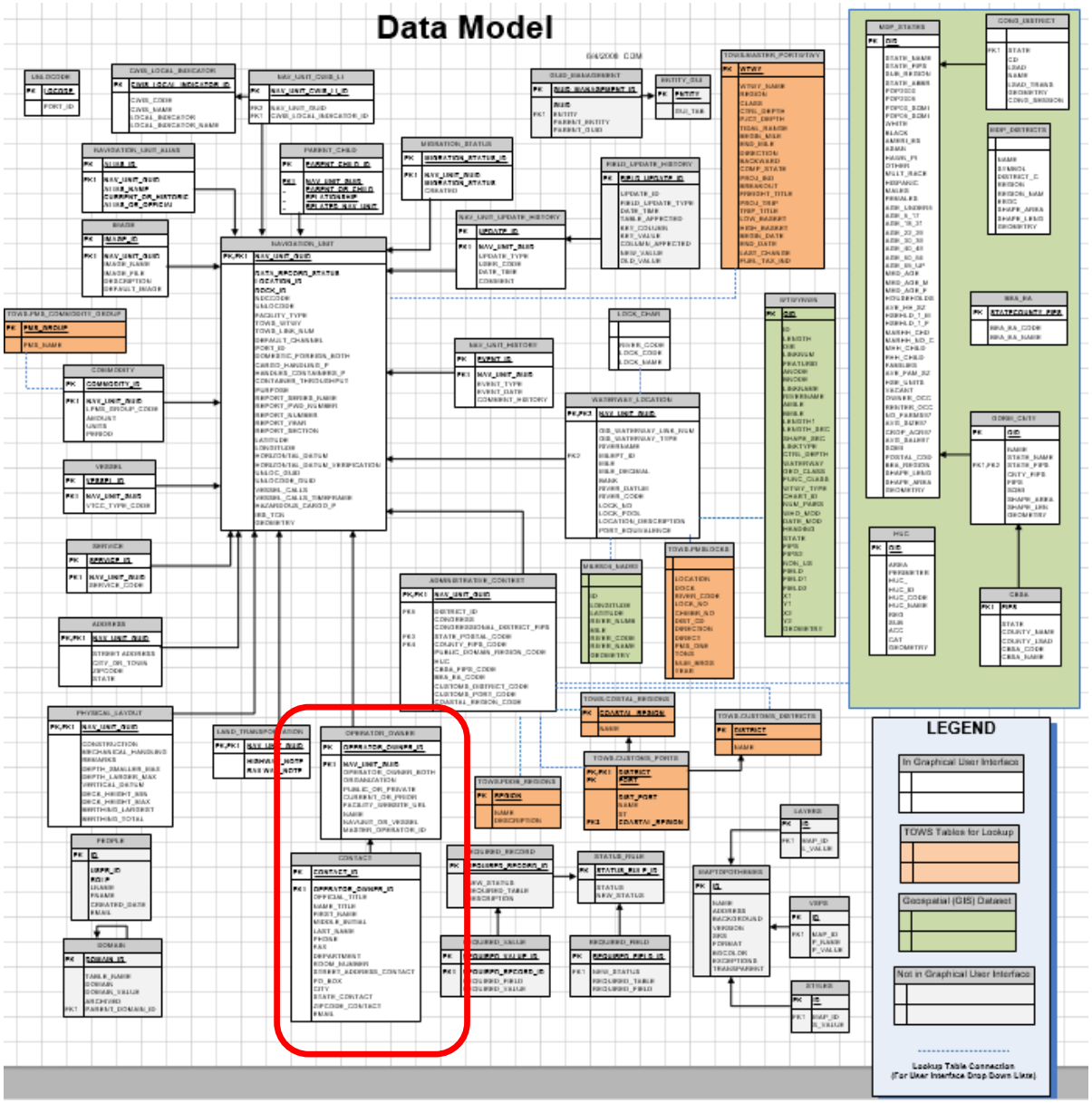
2. **Functional model** – Tasks and functions the application needs to carry out

3. **Behavioral model** – States the application will be in and transition among

Software requirements specifications documents help support:

- Validation
  - "Did they build the right application?"
  - In large complex applications it is easy to lose sight of the main goal ?
  - Does the application/system provide the solution for the intended problem?
- Verification
  - "Did they build the application right?"
  - Applications can be built that do not match the original specifications
    - ***Often not designed/developed with security requirements in mind…***
  - Determines if the application accurately represent and meets the specifications
  - Ensures that the specifications were met properly

1. Informational Model

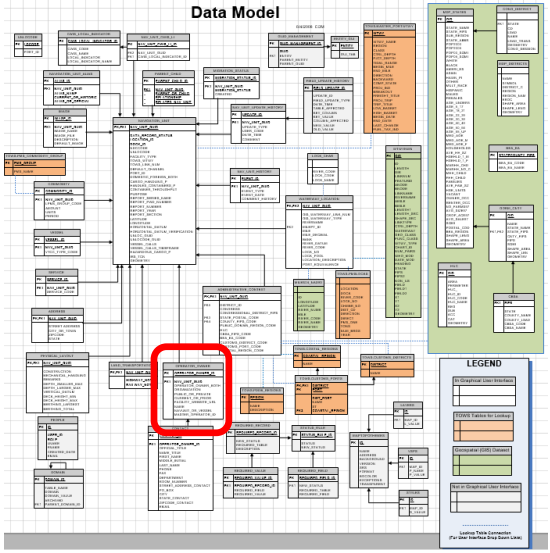


OPERATOR_OWNER	
PK	**OPERATOR_OWNER_ID**
FK1	**NAV_UNIT_GUID** OPERATOR_OWNER_BOTH ORGANIZATION PUBLIC_OR_PRIVATE CURRENT_OR_PRIOR FACILITY_WEBSITE_URL NAME NAVUNIT_OR_VESSEL MASTER_OPERATOR_ID

CONTACT	
PK	**CONTACT_ID**
FK1	**OPERATOR_OWNER_ID** OFFICIAL_TITLE NAME_TITLE FIRST_NAME MIDDLE_INITIAL LAST_NAME PHONE FAX DEPARTMENT ROOM_NUMBER STREET_ADDRESS_CONTACT PO_BOX CITY STATE_CONTACT ZIPCODE_CONTACT EMAIL

Informational Model

Entity-Relational or
UML Data Model



Entity

OPERATOR_OWNER	
PK	OPERATOR_OWNER_ID
FK1	NAV_UNIT_GUID
	OPERATOR_OWNER_BOTH
	ORGANIZATION
	PUBLIC_OR_PRIVATE
	CURRENT_OR_PRIOR
	FACILITY_WEBSITE_URL
	NAME
	NAVUNIT_OR_VESSEL
	MASTER_OPERATOR_ID

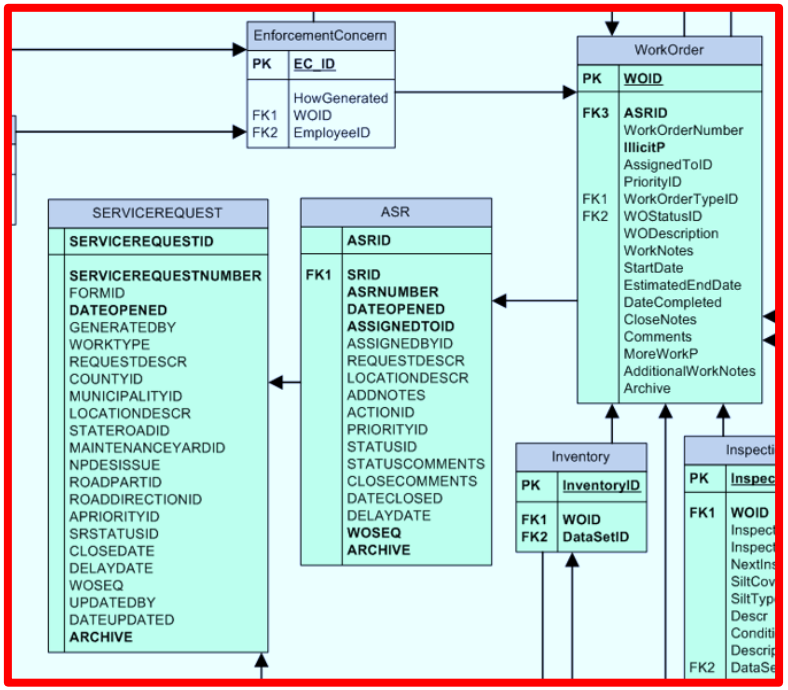
Data dictionary

Widget Type	Widget Name	Table	Column	Data Type	Source Table	Source Column	Source Domain
Data Grid	Owner/Operator/Both	OPERATOR_OWNER	OPERATOR_OWNER_BOTH	VARCHAR2	FILEMAKER	OPERATOR1 & OWNER	DOMAIN
Data Grid	Organization	OPERATOR_OWNER	ORGANIZATION	VARCHAR2	FILEMAKER	ORGANIZATION	
Data Grid	Public or Private	OPERATOR_OWNER	PUBLIC_OR_PRIVATE	VARCHAR2	FILEMAKER	OWNERSHIP	DOMAIN
Data Grid	Current or Prior	OPERATOR_OWNER	CURRENT_OR_PRIOR	VARCHAR2	Derived		DOMAIN
Data Grid	Facility Web site	OPERATOR_OWNER	FACILITY_WEBSITE_URL	VARCHAR2	FILEMAKER	URL_OF_FACILITY	
Data Grid	Name	OPERATOR_OWNER	NAME	VARCHAR2	FILEMAKER	OPERATOR1 / OWNER / PRIOR_OWNER	

Verification

Did they build the application right?

Does it match the data model?



Functional model

Functional Requirements

for

Sewer Outage Notification Application

Version 1.3 Final

Prepared by Jennifer Mattie and David Lanter Ph.D.

July 6, 2005

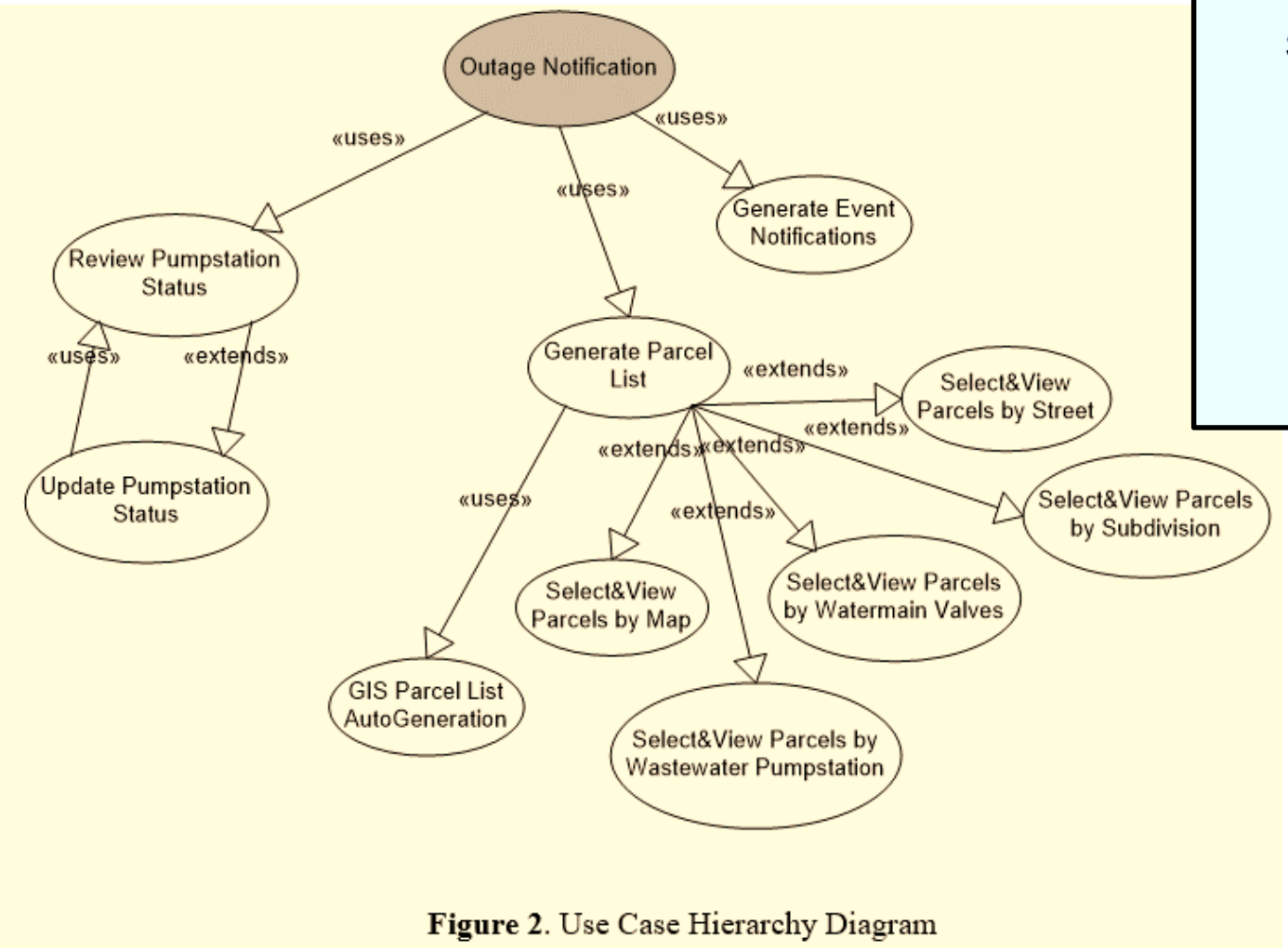


Figure 2. Use Case Hierarchy Diagram

Validation

Did they build the right application?

2. Functional model

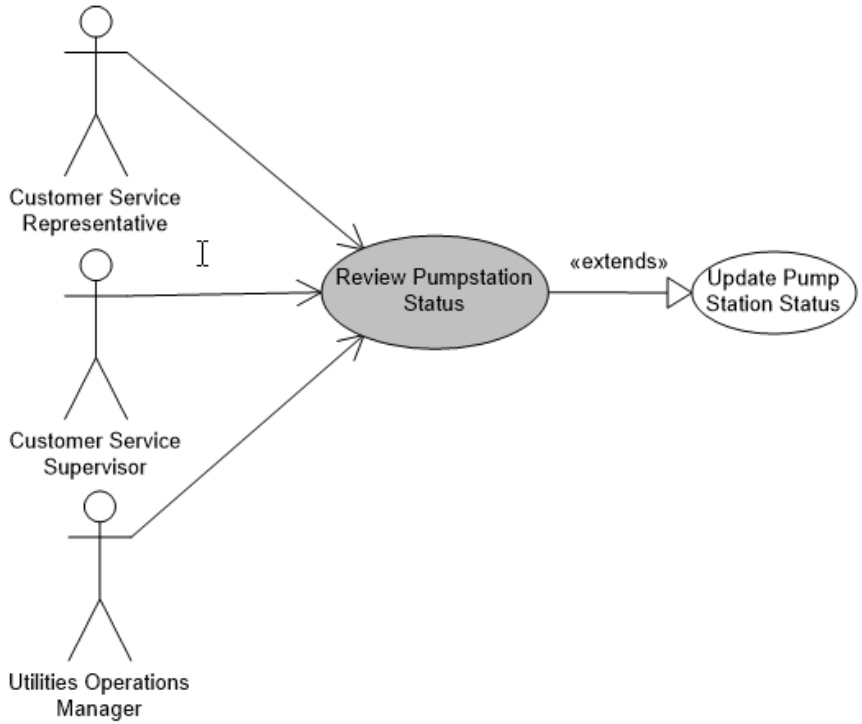


Figure 4. Review Pump Station Status Use Case Diagram

Use Case ID:	1		
Use Case Name:	**Review Pumpstation Status**		
Iteration:	Focused		
Created By:	Jennifer Mattie	Last Updated By:	David Lanter
Date Created:	6-17-2005	Date Last Updated:	7-6-2005
Actor:	Customer Service Representative (CSR) Customer Service Supervisor (CSS) Utilities Operations Manager (UOM)		
Description:	The user (CSR, CSS or UOM) confirms that the pump stations' statuses are up-to-date, before generating an outage event notification list.		
Triggers:	Outage event has occurred or is planned.		
Preconditions:	▪ Up to date pump station GIS feature class dataset with current pump station status values exist are presented to user within GIS application's map user interface. ▪ Parcel GIS feature class dataset must exist and presented to user within GIS application's map user interface. ▪ GIS Data Server online ▪ GIS Web Server online		
Postconditions:	None		
Priority:	Unknown		
Frequency of Use:	Moderate		
Normal Course of Events:	1. User receives information that an outage has occurred, or is planned. 2. User invokes the GIS Outage Notification application. 3. User reviews display of pump stations' statuses on GIS' application's map. 4. User confirms that the pump stations' statuses are up-to-date in the GIS.		
Alternative Courses:	3a. User reviews display of pump station's statuses in pump station status list		
Exceptions:	If the CSR or CSS determines that the pump stations' statuses are not up-to-date, they will notify the UOM responsible for updating the pump station statuses.		
Extensions:	Use Case 2 – Update Pumpstation Status		
Includes:	None		
Related Business Rules:	None		
Special Requirements:	None		
Assumptions:	User provided with GUI control to invoke this use case.		
Notes and Issues:	▪ It is not clear how User knows for certain that the pump stations' statuses are correct in the GIS. ▪ SCADA or a real-time data feedback system is required to assure that pump stations' statuses are all correct and up-to-date. ▪ CSR or CSS must work through the UOM to assure that the status of the pumpstations are correct.		

Validation

Did they build the right application?

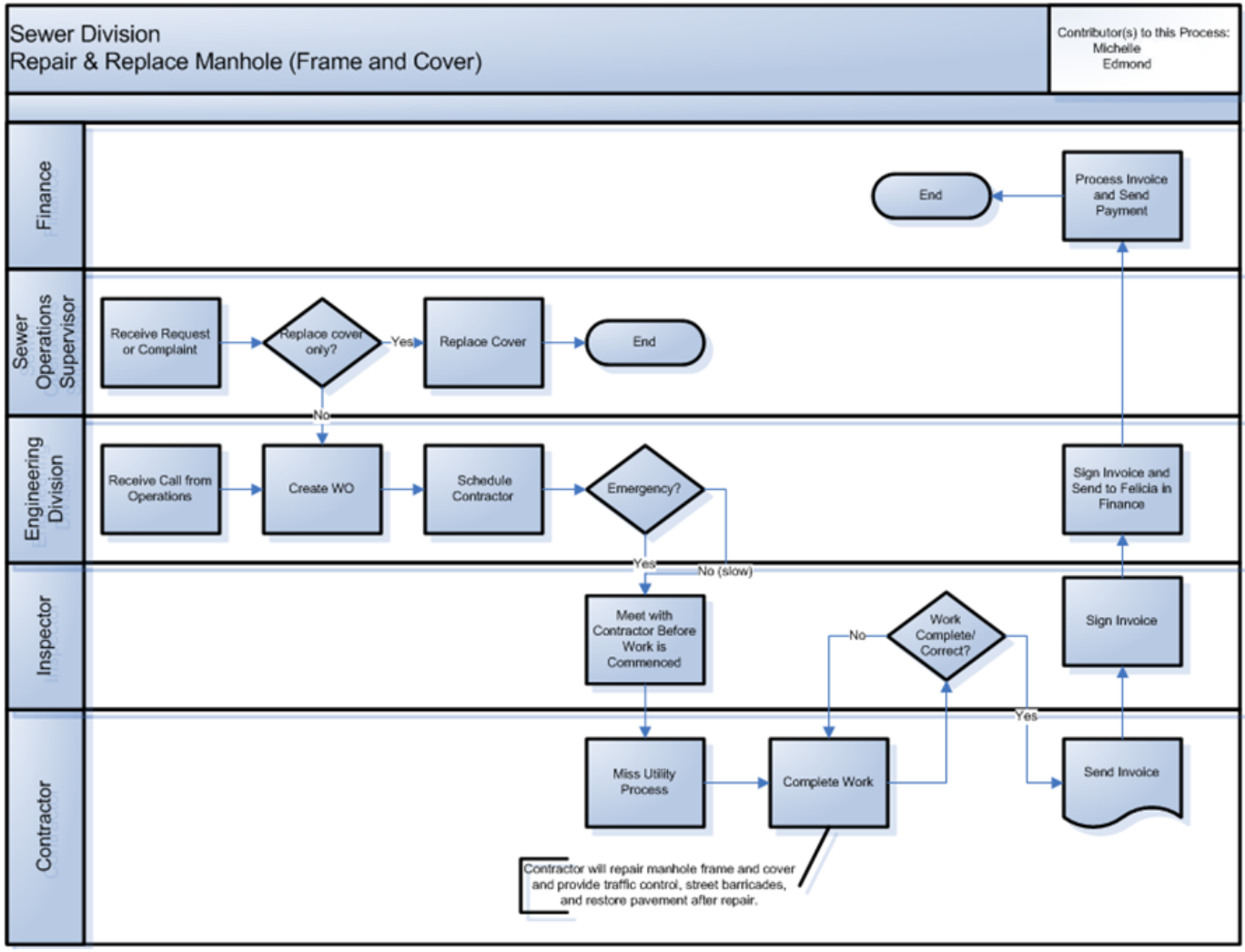
3. Behavioral models – swim lane model

Validation

"Did they build the right application?"

Verification

"Did they build the application right?"



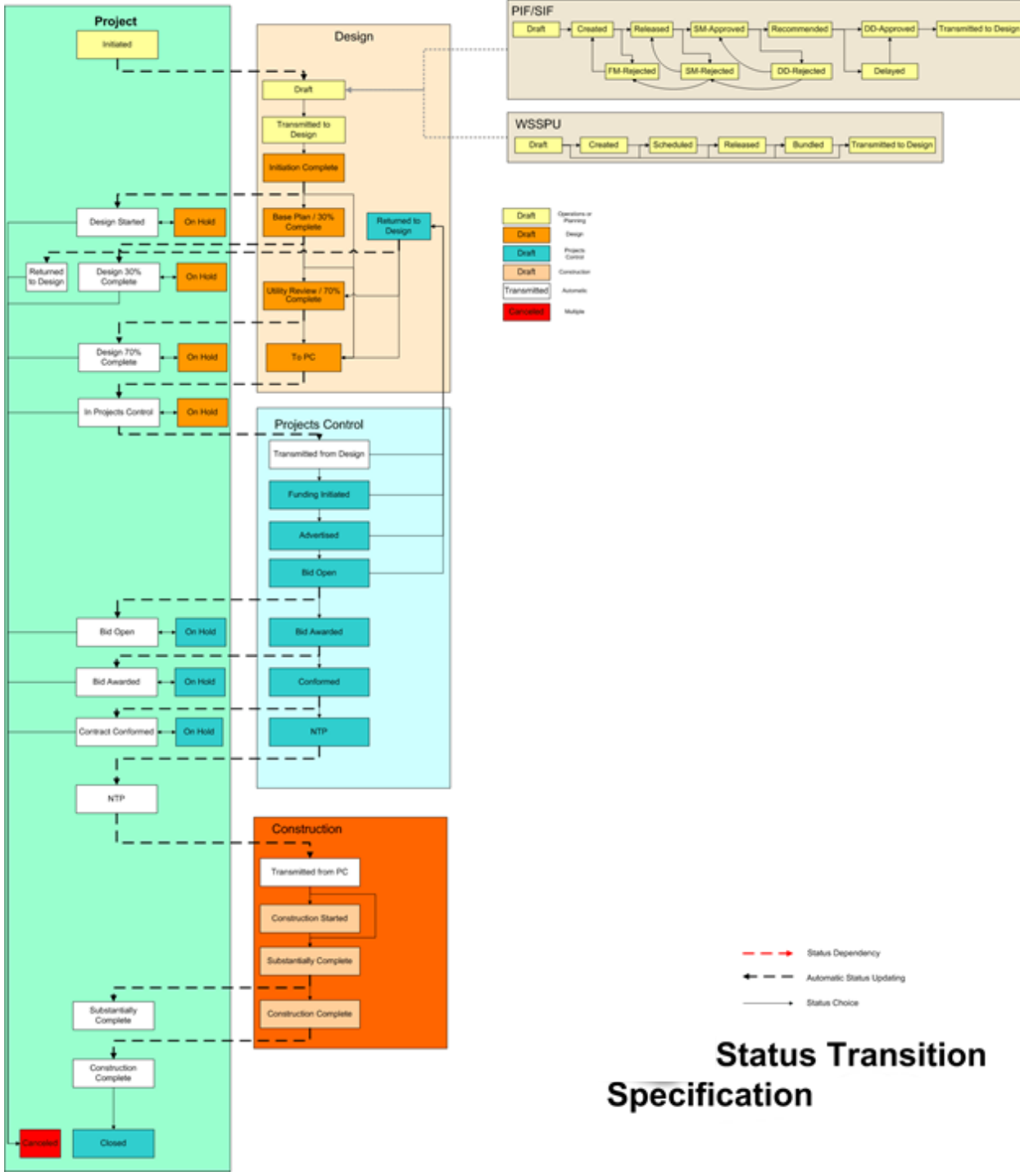
3. Behavioral model

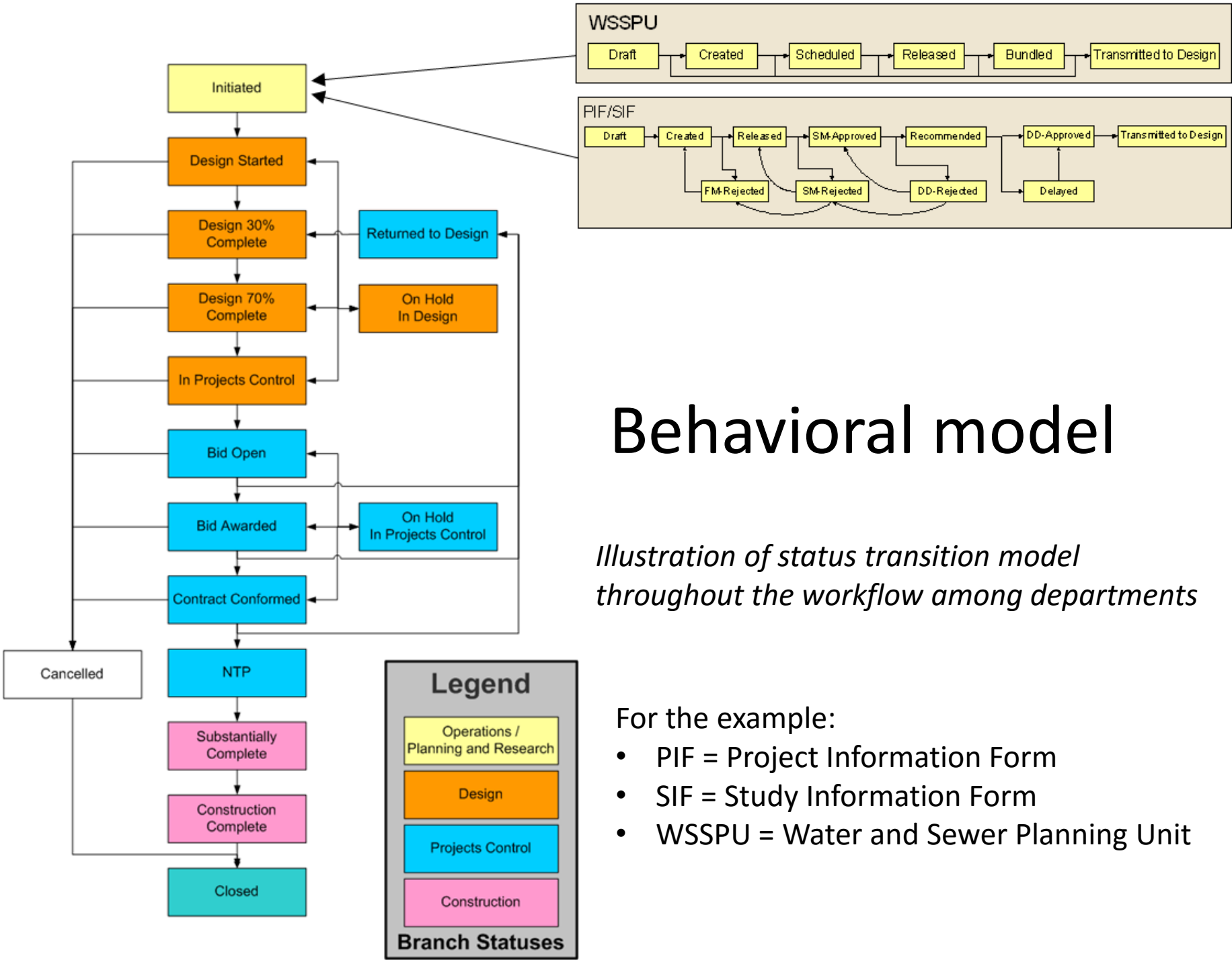
Validation

"Did they build the right application?"

Verification

"Did they build the application right?"





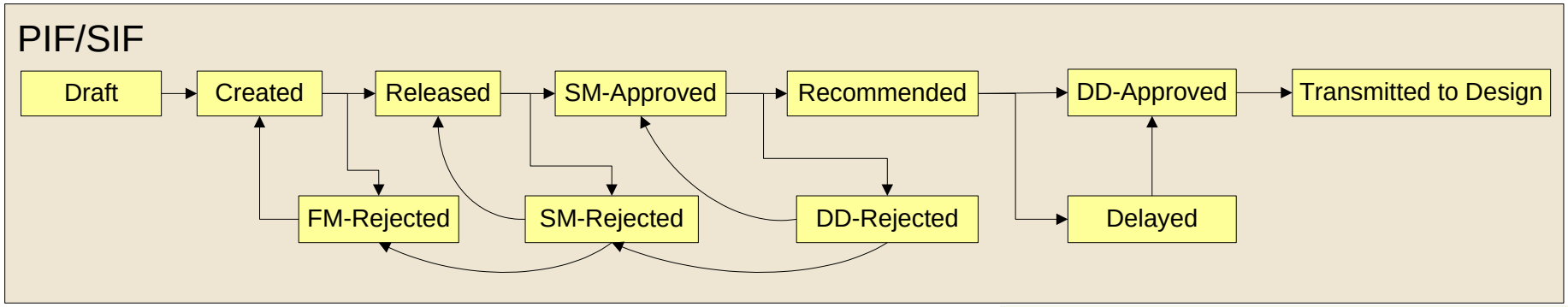
Behavioral model

Illustration of status transition model throughout the workflow among departments

For the example:

- PIF = Project Information Form
- SIF = Study Information Form
- WSSPU = Water and Sewer Planning Unit

3. Behavioral model – workflow/status transition model



For the example:

- PIF = Project Information Form
- SIF = Study Information Form

PIF Status Summary

	Status	Count
1	Not Started	0
2	Draft	64
3	Created	10
4	Released	587
5	FM-Rejected	29
6	SM-Approved	0
7	SM-Rejected	0
8	Recommended	0
9	DD-Approved	6
10	DD-Rejected	0
11	Delayed	0
12	Transmitted to Design	2
	TOTAL	698

SDLC and Security

Requirements analysis

- *Informational, functional, behavioral, and performance specifications…*

+ CIA risk assessment, + Risk-level acceptance,…

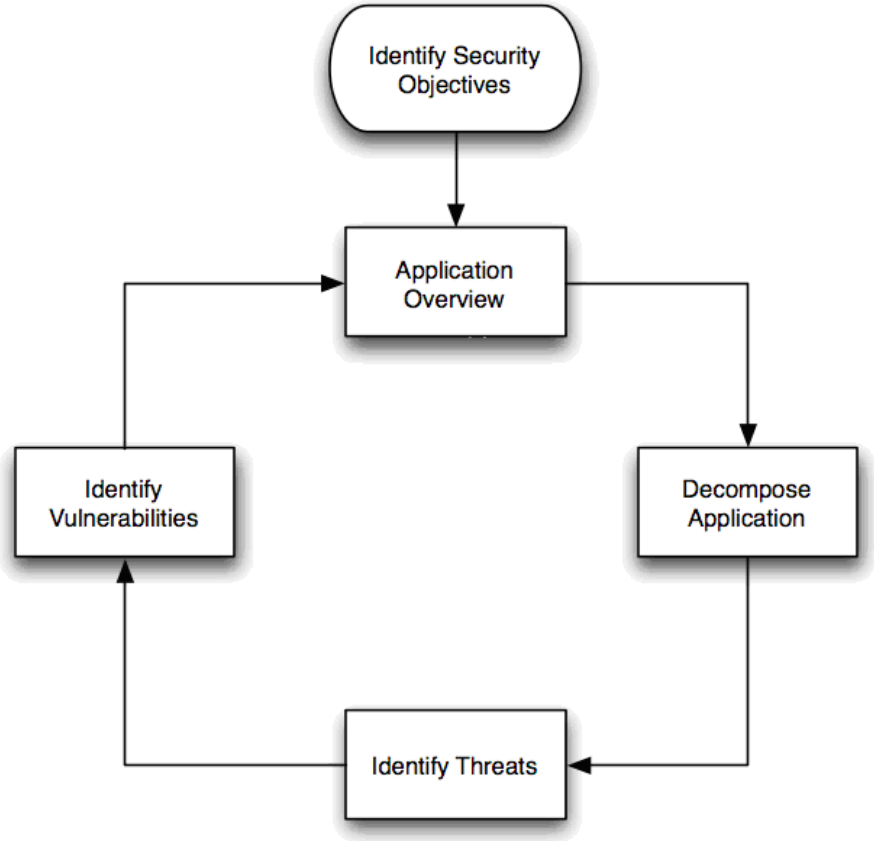
Design

- *Data models and data dictionary, work process and status transition models, input/output models, data flow models, flow of control models…*

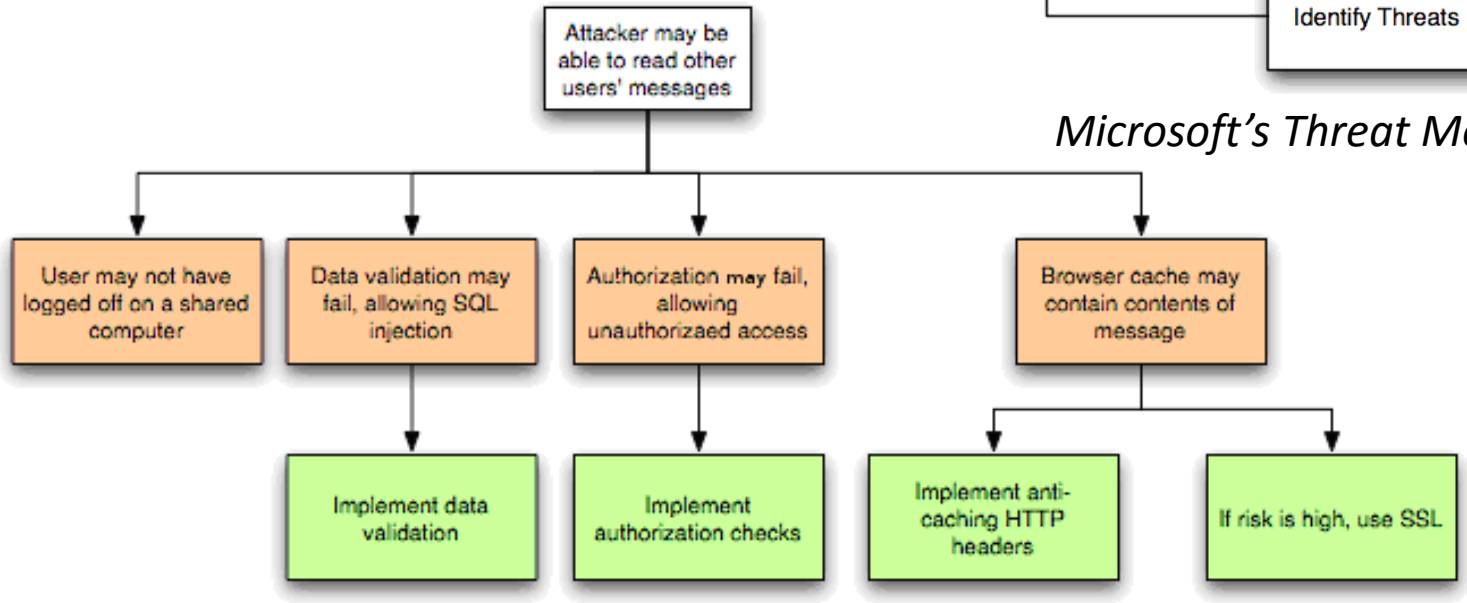
+ Threat modeling, + Attack surface analysis,…

SDLC Design Security

Threat modeling *is a systematic approach for understanding how different threats could be realized and a successful attack could take place*



Microsoft's Threat Modeling Process

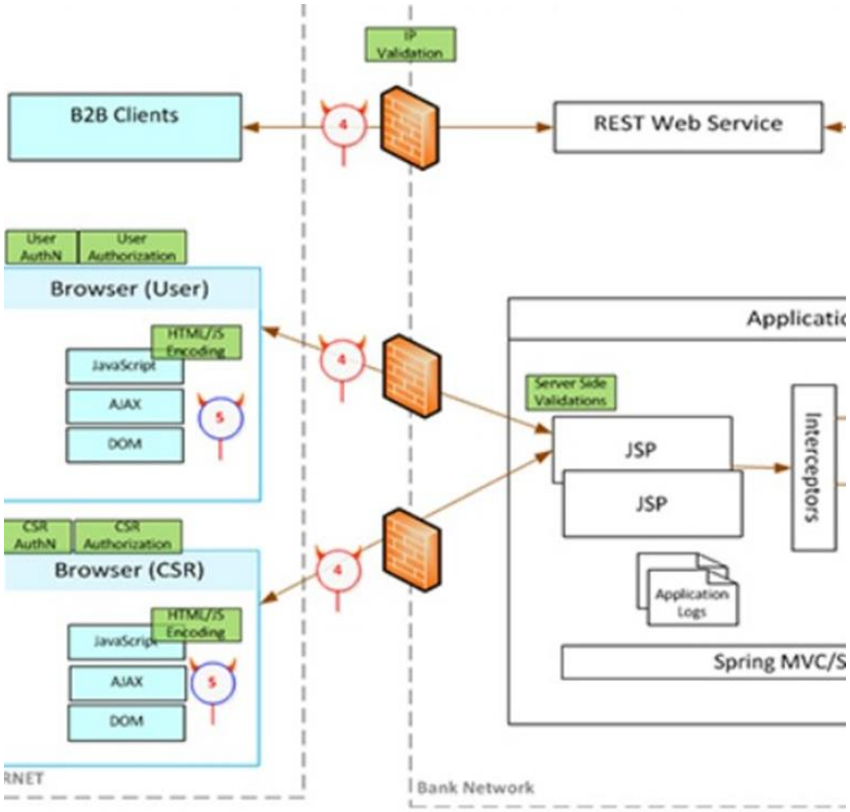


SDLC Design Security

Attack surface *is what is available to be used by an attacker against the application itself*

Goal of attack surface analysis is to identify and reduce the amount of code and functionality accessible to untrusted users

Development team should reduce the attack surface as much as possible to remove "resources" that can be used as avenues for the attacker to use



MITRE's Common Application Vulnerabilities

Secure | https://cwe.mitre.org/data/definitions/699.html

Common Weakness Enumeration
A Community-Developed List of Software Weakness Types

CWE and SANS Institute TOP 25 MOST DANGEROUS SOFTWARE ERRORS

Home > CWE List > CWE- Individual Dictionary Definition (3.0) ID Lookup: Go

Home | About | CWE List | Scoring | Community | News | Search

CWE VIEW: Development Concepts

View ID: 699
Type: Graph
Status: Incomplete

Objective

This view organizes weaknesses around concepts that are frequently used or encountered in software development. Accordingly, this view can align closely with the perspectives of developers, educators, and assessment vendors. It borrows heavily from the organizational structure used by Seven Pernicious Kingdoms, but it also provides a variety of other categories that are intended to simplify navigation, browsing, and mapping.

Audience

Stakeholder	Description
Assessment Vendors	
Software Developers	
Educators	

Relationships

699 - Development Concepts

- Configuration - *(16)*
- Data Processing Errors - *(19)*
- Pathname Traversal and Equivalence Errors - *(21)*
- Numeric Errors - *(189)*
- 7PK - Security Features - *(254)*
- 7PK - Time and State - *(361)*
- Error Conditions, Return Values, Status Codes - *(389)*
- Resource Management Errors - *(399)*
- Channel and Path Errors - *(417)*
- Handler Errors - *(429)*
- Behavioral Problems - *(438)*
- Business Logic Errors - *(840)*
- Web Problems - *(442)*
- User Interface Security Issues - *(355)*
- Initialization and Cleanup Errors - *(452)*
- Pointer Issues - *(465)*
- Mobile Code Issues - *(490)*
- Often Misused: Arguments and Parameters - *(559)*
- Expression Issues - *(569)*
- Violation of Secure Design Principles - *(657)*
- Bad Coding Practices - *(1006)*

- **Development Concepts**
  - Configuration - *(16)*
  - Data Processing Errors - *(19)*
  - Pathname Traversal and Equivalence Errors - *(21)*
  - Numeric Errors - *(189)*
  - 7PK - Security Features - *(254)*
  - 7PK - Time and State - *(361)*
  - Error Conditions, Return Values, Status Codes - *(389)*
  - Resource Management Errors - *(399)*
  - Channel and Path Errors - *(417)*
  - Handler Errors - *(429)*
  - Behavioral Problems - *(438)*
  - Business Logic Errors - *(840)*
  - Web Problems - *(442)*
  - User Interface Security Issues - *(355)*
  - Initialization and Cleanup Errors - *(452)*
  - Pointer Issues - *(465)*
  - Mobile Code Issues - *(490)*
  - Often Misused: Arguments and Parameters - *(559)*
  - Expression Issues - *(569)*
  - Violation of Secure Design Principles - *(657)*
  - Bad Coding Practices - *(1006)*

MITRE's Common Weakness Enumeration

Login | Cre

Find Training | Live Training | Online Training | Programs

CWE/SANS TOP 25 Most Dangerous Software Errors

Insecure Interaction Between Components

These weaknesses are related to insecure ways in which data is sent and received between separate components, modules, programs, processes, threads, or systems.

CWE ID	Name
CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')
CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')
CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')
CWE-434	Unrestricted Upload of File with Dangerous Type
CWE-352	Cross-Site Request Forgery (CSRF)
CWE-601	URL Redirection to Untrusted Site ('Open Redirect')

Porous Defenses

The weaknesses in this category are related to defensive techniques that are often misused, abused, or just plain ignored.

CWE ID	Name
CWE-306	Missing Authentication for Critical Function
CWE-862	Missing Authorization
CWE-798	Use of Hard-coded Credentials
CWE-311	Missing Encryption of Sensitive Data
CWE-807	Reliance on Untrusted Inputs in a Security Decision
CWE-250	Execution with Unnecessary Privileges
CWE-863	Incorrect Authorization
CWE-732	Incorrect Permission Assignment for Critical Resource
CWE-327	Use of a Broken or Risky Cryptographic Algorithm
CWE-307	Improper Restriction of Excessive Authentication Attempts
CWE-759	Use of a One-Way Hash without a Salt

Risky Resource Management

The weaknesses in this category are related to ways in which software does not properly manage the creation, usage, transfer, or destruction of important system resources.

CWE ID	Name
CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
CWE-494	Download of Code Without Integrity Check
CWE-829	Inclusion of Functionality from Untrusted Control Sphere
CWE-676	Use of Potentially Dangerous Function
CWE-131	Incorrect Calculation of Buffer Size
CWE-134	Uncontrolled Format String
CWE-190	Integer Overflow or Wraparound

OWASP

OWASP Top 10 - 2017

The Ten Most Critical Web Application Security Risks

https://owasp.org

SDLC and Security

Requirements analysis

- *Informational, functional, behavioral, and performance specifications…*

+ CIA risk assessment, + Risk-level acceptance,…

Design

- *Data models and data dictionary, work process and status transition models, input/output models, data flow models, flow of control models…*

+ Threat modeling, + Attack surface analysis,…

Develop (“*make*”) / Implement (“*buy*”)

- *Source code control system, code reviews, daily builds, automated CASE tools…*

+ Developer security training, + Static analysis, + Secure code repositories,…

Curricula

Secure Software Development

Secure Software Development Curriculum		
	Course	**Certification**
Level 1	DEV522: Defending Web Applications Security Essentials	GWEB
	DEV531: Defending Mobile Applications Security Essentials	
	DEV534: Secure DevOps: A Practical Introduction	
Level 2	DEV541: Secure Coding in Java/JEE: Developing Defensible Applications	GSSP-JAVA
	DEV543: Secure Coding in C & C++	
	DEV544: Secure Coding in .NET: Developing Defensible Applications	GSSP-.NET
Specialty Courses	SEC542: Web App Penetration Testing and Ethical Hacking	GWAPT
	SEC642: Advanced Web App Penetration Testing, Ethical Hacking, and Exploitation Techniques	
	HST: Certified Secure Software Lifecycle Professional (CSSLP®) CBK® Training Seminar	

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
Deutsch
Español
Edit links

Article Talk

List of tools for static code analysis

From Wikipedia, the free encyclopedia

This article **needs additional cita**
(Learn how and when to remove this ter

This is a list of tools for static code analysis.

Contents [hide]

Language [edit]

Multi-language [edit]

- Axivion Bauhaus Suite – A tool for Ada, C, C++, C#, and Java code tha
- BlueOptima – Coding Effort Analytics objectively measure the producti ensure the maintainability and stability of a code base.
- CAST Application Intelligence Platform – Detailed, audience-specific d major databases.

Code Repositories

Name	Manager	Established	Server side: all free software	Client side: all-free JS code
Alioth	Debian Project	2003	Yes	Yes
Assembla	Assembla, Inc	2005	No	Unknown
Betavine	Vodafone	2007	No	Unknown
Bitbucket	Atlassian	2008	No	No
CloudForge	CollabNet	2000	No	Unknown
CodePlex	Microsoft	2006-05	No	Unknown
GitHub	GitHub, Inc	2008-04	No	No
GitLab	GitLab B.V.	2011-09[5]	Yes	Yes[6]
Gna!	Unknown	2004-01	Yes	Yes
GNU Savannah	Savannah Administration	2001-01	Yes	Yes
Kallithea	SFC	2014[9]	Yes	Yes
Launchpad	Canonical	2004	Yes	No
OSDN	OSDN K.K.	2002-04	Unknown	Yes
Ourproject.org	Comunes Collective	2002	Yes	No
OW2 Consortium	OW2 Consortium	Unknown	Unknown	No
Rosetta Code	Unknown	2007	Unknown	Unknown
SEUL	Unknown	1997-05	Unknown	No
SourceForge	BizX LLC	1999-11	Yes[12][13]	Yes
Tigris.org	(community)	2000	Unknown	No
Team Foundation Server	Microsoft	2012-2005[16]	Unknown	No
Visual Studio Team Services	Microsoft	2012[17]	Yes	No

SDLC and Security

Requirements analysis

- *Informational, functional, behavioral, and performance specifications…*

+ CIA risk assessment, + Risk-level acceptance,…

Design

- *Data models and data dictionary, work process and status transition models, input/output models, data flow models, flow of control models…*

+ Threat modeling, + Attack surface analysis,…

Develop (“*make*”) / Implement (“*buy*”)

- *Source code control system, code reviews, daily builds, automated CASE tools…*

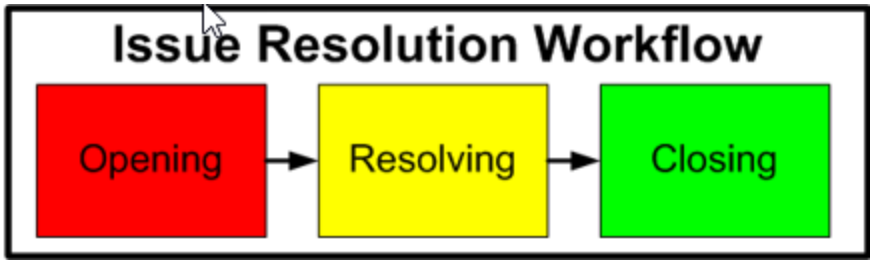
+ Developer security training, + Static analysis, + Secure code repositories,…

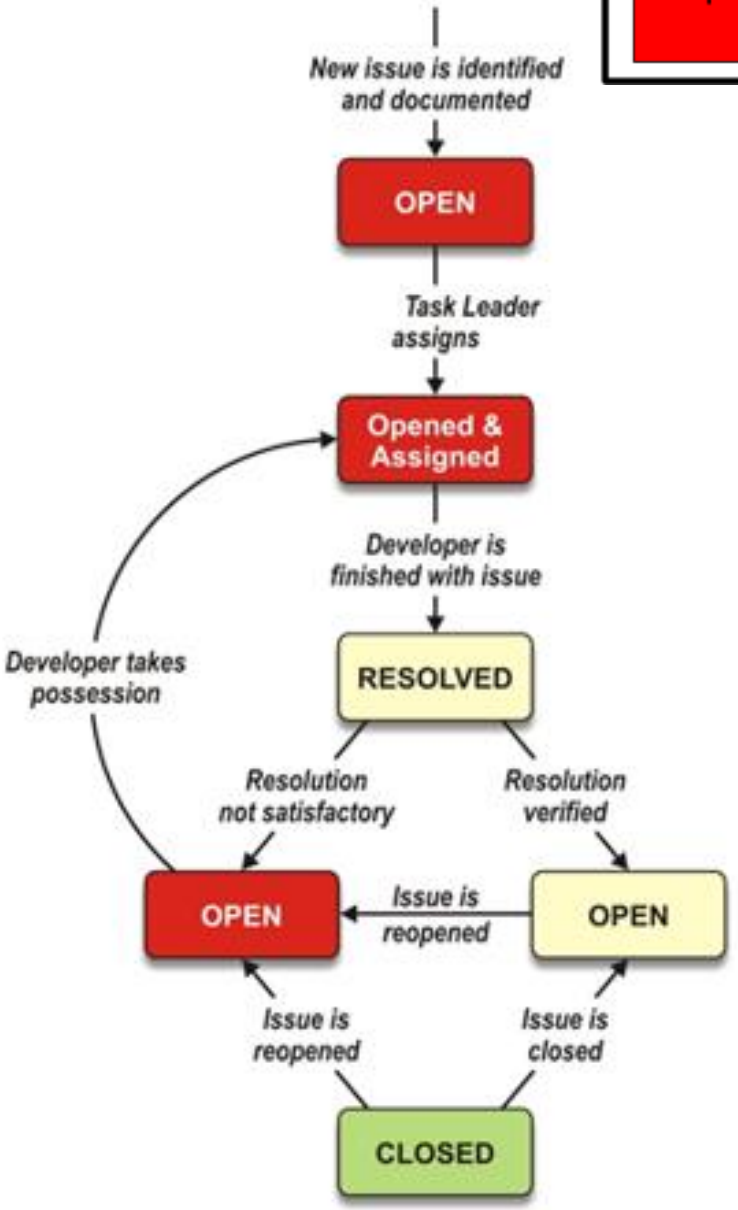
Testing/Validation

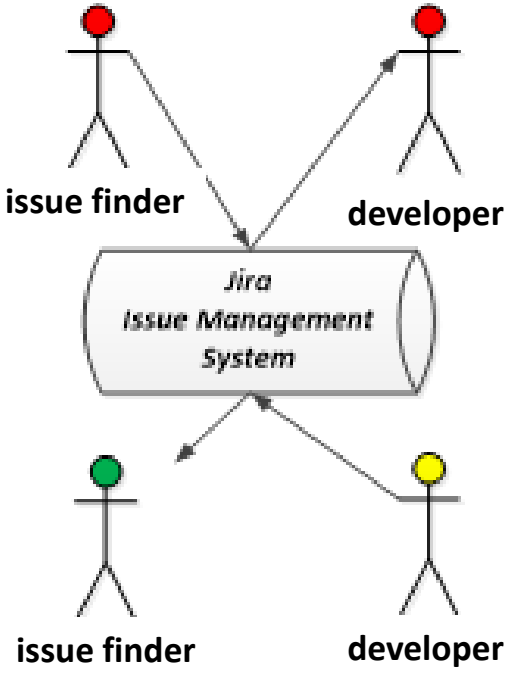
- *Unit testing and integration testing (daily builds), manual and regression testing, user acceptance testing*

+ Dynamic analysis, + Fuzzing,…

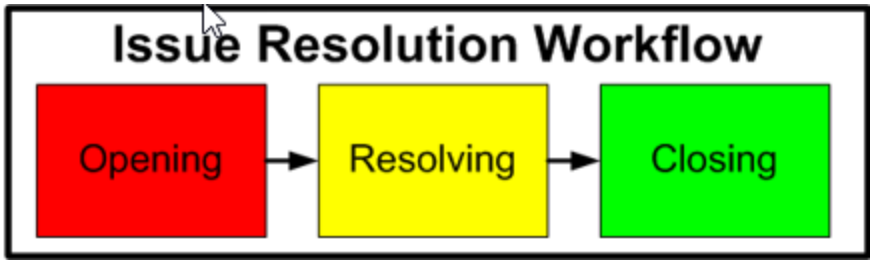
Testing/validation







Testing/validation



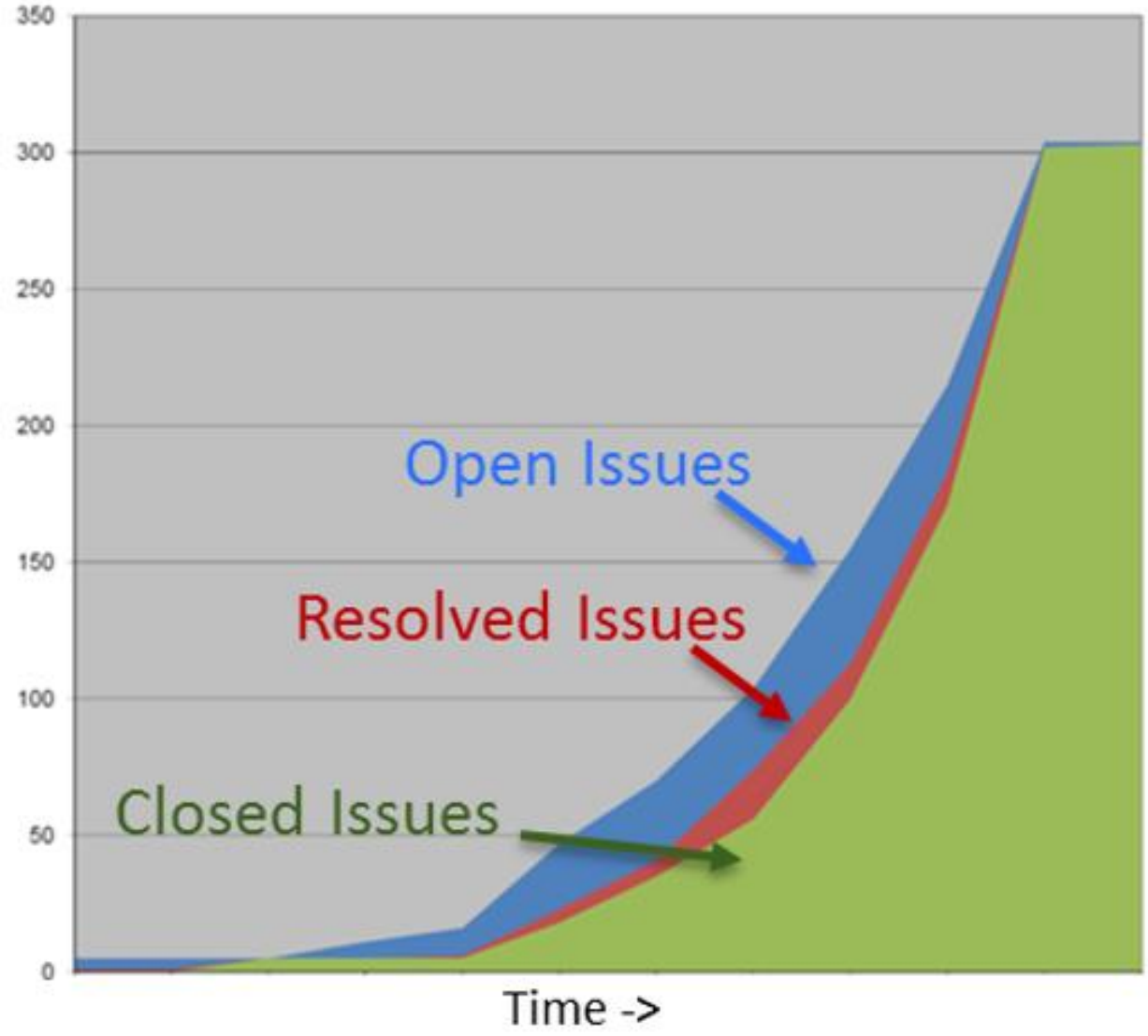


Figure 1. Magic Quadrant for Application Security Testing

Source: Gartner (March 2018)

Application security testing tool providers

Burp Scanner Sample Report

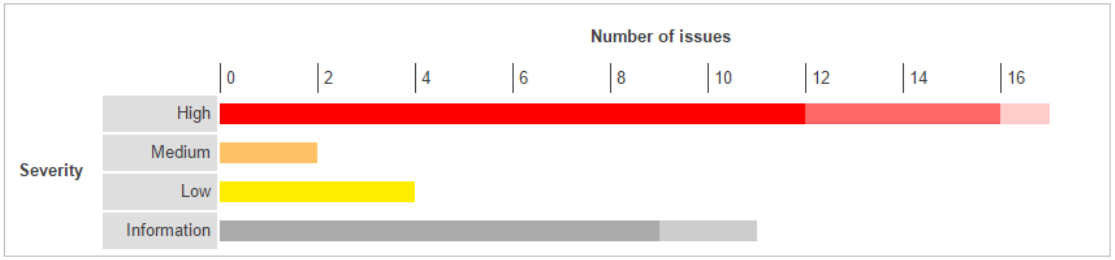
BURPSUITE PROFESSIONAL

Summary

The table below shows the numbers of issues identified in different categories. Issues are classified according to severity as High, Medium, Low or Information. This reflects the likely impact of each issue for a typical organization. Issues are also classified according to confidence as Certain, Firm or Tentative. This reflects the inherent reliability of the technique that was used to identify the issue.

Severity	Confidence			
	Certain	Firm	Tentative	Total
High	12	4	1	17
Medium	0	2	0	2
Low	4	0	0	4
Information	9	2	0	11

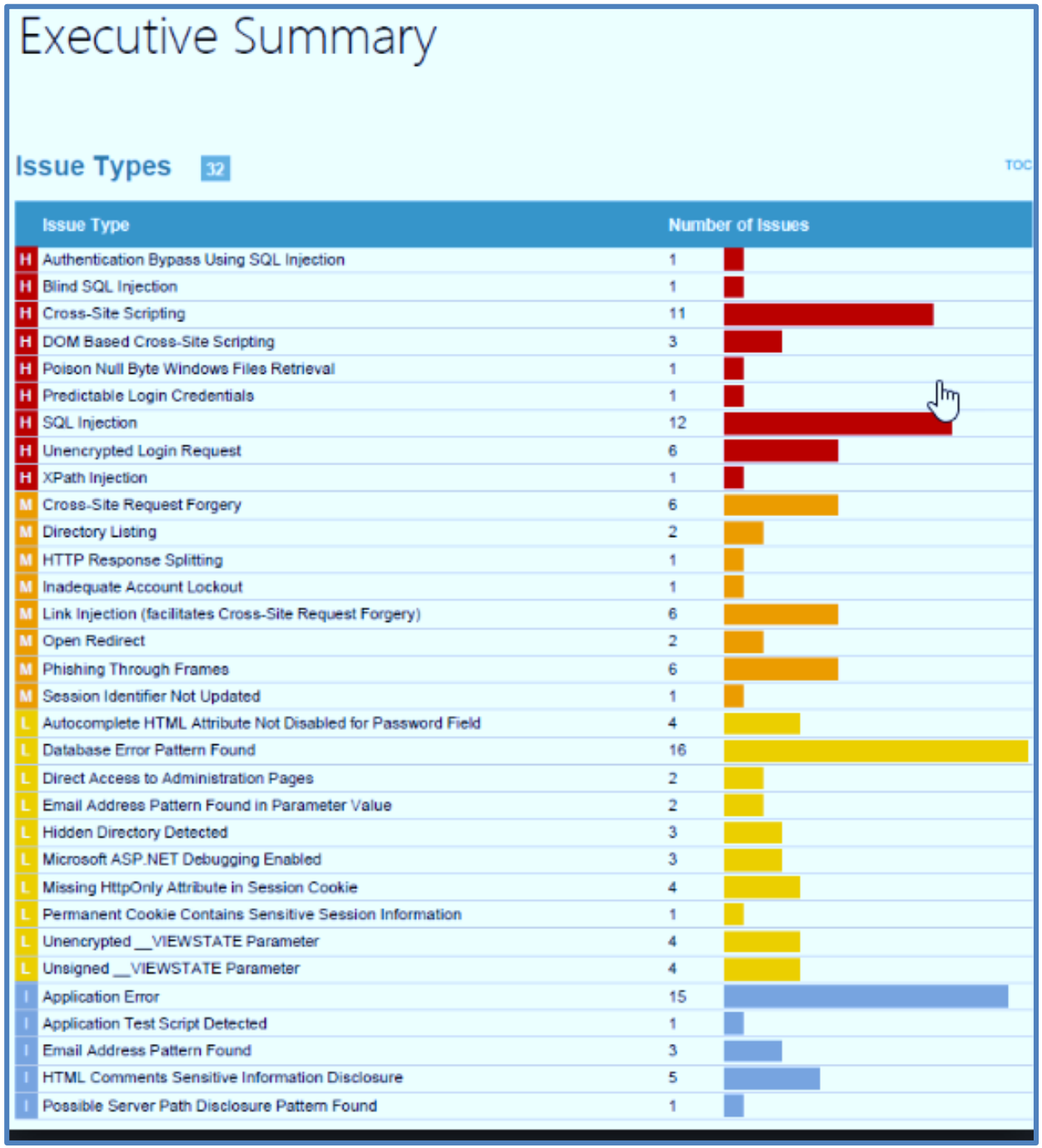
The chart below shows the aggregated numbers of issues identified in each category. Solid colored bars represent issues with a confidence level of Certain, and the bars fade as the confidence level falls.



Contents

Dynamic code testing result reports

- Applications should not accepted until ***all high and medium issues resolved!***



Executive Summary

Issue Types 32

TOC

	Issue Type	Number of Issues
H	Authentication Bypass Using SQL Injection	1
H	Blind SQL Injection	1
H	Cross-Site Scripting	11
H	DOM Based Cross-Site Scripting	3
H	Poison Null Byte Windows Files Retrieval	1
H	Predictable Login Credentials	1
H	SQL Injection	12
H	Unencrypted Login Request	6
H	XPath Injection	1
M	Cross-Site Request Forgery	6
M	Directory Listing	2
M	HTTP Response Splitting	1
M	Inadequate Account Lockout	1
M	Link Injection (facilitates Cross-Site Request Forgery)	6
M	Open Redirect	2
M	Phishing Through Frames	6
M	Session Identifier Not Updated	1
L	Autocomplete HTML Attribute Not Disabled for Password Field	4
L	Database Error Pattern Found	16
L	Direct Access to Administration Pages	2
L	Email Address Pattern Found in Parameter Value	2
L	Hidden Directory Detected	3
L	Microsoft ASP.NET Debugging Enabled	3
L	Missing HttpOnly Attribute in Session Cookie	4
L	Permanent Cookie Contains Sensitive Session Information	1
L	Unencrypted __VIEWSTATE Parameter	4
L	Unsigned __VIEWSTATE Parameter	4
I	Application Error	15
I	Application Test Script Detected	1
I	Email Address Pattern Found	3
I	HTML Comments Sensitive Information Disclosure	5
I	Possible Server Path Disclosure Pattern Found	1

Dynamic code testing result reports

- Applications should not accepted until *all high and medium issues resolved!*

SDLC and Security

Requirements analysis

- *Informational, functional, behavioral, and performance specifications…*

+ CIA risk assessment, + Risk-level acceptance,…

Design

- *Data models and data dictionary, work process and status transition models, input/output models, data flow models, flow of control models…*

+ Threat modeling, + Attack surface analysis,…

Develop ("*make*") / Implement ("*buy*")

- *Source code control system, code reviews, daily builds, automated CASE tools…*

+ Developer security training, + Static analysis, + Secure code repositories,…

Testing/Validation

- *Unit testing and integration testing (daily builds), manual and regression testing, user acceptance testing*

+ Dynamic analysis, + Fuzzing,…

Release/Maintenance

- *Release testing*

+ Separation of duties, +Change management, +Operational practices…

Separation of Duties

Different environments (development, testing, and production) should be separated, without overlapping access to code, applications and systems

The access and ability of developers to modify application code make them the "most powerful" insider threats and vulnerabilities to information systems

- Developers should not have access to modify code used in production
- Code should be tested, submitted to a library, and then sent to the production environment

Releases/Maintenance

- Commercial Off The Shelf (COTS) products and Open Source products should have their security patches
- Installation programs should be removed from production
- File and program settings and privileges should be reviewed

Operational concerns

- Commercial Off The Shelf (COTS) software sources of risk
- Open source libraries sources of risk
- Operational Practices
  - System Security Plan (SSP) updates
  - Contingency Plan (BCP/DRP) updates
  - Awareness and training updates
  - Documentation updates

Operational Practices

- Support training classes
- User administration and access privileges
- Backup and restoration
  - Data, applications, configurations, restart instructions and procedures
  - Performing backups: How often? In which ways?
  - Performing backups
  - Offsite storage
  - Testing restoration

- Ensure implementation of only approved and accredited systems

- Cryptography keys
  - Generation and Use
  - Protection and storage
- Audit logs
  - How collected?
  - Where stored?
  - How protected?
  - How analyzed?

Operational Assurance Activities

- Review
  - Interdependencies among applications and systems
  - Runtime operation
  - Technical controls
- Verify documentation
  - Of access permissions
  - Is current and accurate
- Verify proper deregistration
  - i.e. removal of users and privileges
- Is availability and distribution of output products secure?
- Are software & hardware licenses fulfilled and warrantees in place?

Other topics: Disposal

- Storage and protection of cryptographic keys
- Legal requirements of records retention
- Archiving federal information
- Sanitize media

Test Taking Tip

Focus on addressing each question individually

- As you take the test, if you don't know an answer, don't obsess over it
- Answer the best way you can or skip over the question and come back to it after you've answered other questions

Quiz

Agenda

- ✓ Introduction
- ✓ Software development life cycle (SDLC)
- ✓ SDLC and security
- ✓ Test taking tip
- ✓ Quiz